LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-JRNL-568309

Message Passing Interface and Multithreading Hybrid for Parallel Molecular Docking of Large Databases on Petascale High Performance Computing Machines

X. Zhang, S. E. Wong, F. C. Lightstone

August 1, 2012

Journal of Computational Chemistry

Disclaimer

Message Passing Interface and Multithreading Hybrid for Parallel Molecular Docking of Large Databases on Petascale High Performance Computing Machines

Xiaohua Zhang, Sergio E. Wong, and Felice C. Lightstone*

Abstract

A mix of message passing interface (MPI) and multithreading parallel scheme has been implemented in the AutoDock Vina molecular docking program. The resulting program, named VinaLC, has been tested on the petascale high performance computing (HPC) machines at Lawrence Livermore National Laboratory. To exploit the typical cluster-type supercomputers, thousands of docking calculations were dispatched by the master process to run simultaneously on thousands of slave processes, where each docking calculation takes one slave process on one node, and within the node each docking calculation runs via multithreading on multiple CPU cores and shared memory. Input and output of program and the data handling within the program have been carefully designed to deal with large databases and ultimately achieve high performance computing on a large number of CPU cores. Parallel performance analysis of the VinaLC program shows that the code can scale up to more than 15K CPUs with a very low overhead cost of 3.94%. One million flexible compounds docking calculations took only 1.4 hours to finish on about 15K CPUs. The docking accuracy of VinaLC has been validated against the DUD data set by the re-docking of the X-ray ligands and an enrichment study. The re-docking of the X-ray ligands have shown that 64.4% of the top scoring poses with RMSD values under the 2.0 Å cutoff. The program has been demonstrated to have good enrichment performance on 70% targets in the DUD data set. An analysis of the enrichment factors calculated at various percentages of the screening database indicates VinaLC has very good early recovery of actives.

Introduction

Structure-based high throughput virtual screening has been widely applied in the early stage of drug discovery because of its low cost, high efficiency, and some crucial successes in recent years[1-3]. Molecular docking based virtual screening involves a target protein structure, either experimentally solved or computationally modeled, and a library of small molecules that fit into the active site of this target to estimate binding affinity. About 10% of the human genome, containing ~30,000 genes that express proteins able to bind drug-like molecules, is estimated to be druggable[4]. In the past few years the number of protein structures publicly available has increased rapidly, not to mention many other structures held by the pharmaceutical companies. The Research Collaboratory for Structural Biology (RCSB) database[5] currently has more than 80K protein structures deposited in the database, with thousands being added each year. In addition, the chemical space of small molecules is enormous and the number of drug-like compounds has been estimated to be 10^{60}.[6] The ZINC database[7], a free database of commercially available compounds for virtual screening, has more than 30 million entries up to date. Millions of potential drug candidates are required to be screened against the ever-increasing number of the druggable biological targets, which makes it an urgent task to develop fast and accurate screening techniques that can deal with massive amounts of data.

Various docking programs, either open-source (e.g. DOCK[8,9], AutoDock[10,11]) or commercially available (e.g. FlexX[12], GLIDE[13,14], GOLD[15]), have been developed and have evolved in the last few decades. Molecular docking is basically a conformational sampling procedure to identify the best binding pose of a small molecule fitting into a protein target[16]. The conformational sampling procedure is usually based on Monte Carlo simulations, simulated annealing, genetic algorithm, or other sampling methods. The sampled poses are then evaluated with various scoring functions

and ranked according to their scores[17]. The types of scoring functions can be divided into four categories: molecular mechanics force-field method, empirical scoring functions, quantum mechanical method, and knowledge-based potentials[18,19]. Docking programs can employ a combination of different sampling techniques and scoring functions to achieve certain accuracies at various speeds. Most docking programs were originally designed to run on personal computers or small workstations. The scalability of these programs often becomes poor when using a large number of CPU cores. Some commercial programs even set tokens to limit the number of jobs that can be run in parallel.

With the growth power of supercomputers, efforts have been made to parallelize the docking programs and to scale up to a large number of CPU cores. There are several ways to implement a parallel docking program, running on high performance computing (HPC) machines. The first approach is to launch parallel jobs through a job dispatching system, e.g. portable batch system (PBS)[20] is a popular job scheduler on HPC machines. DOVIS 2.0 is a parallel docking program based on AutoDock 4[21] and relies mainly on the job queuing system to run the job in parallel[22]. For scale up to 256 CPUs, DOVIS 2.0 has reasonable efficiency. However, the major bottleneck for this approach is that the communication between individual jobs must use intermediate files, which increases the I/O activity, slows down the calculation, and limits the maximum number of CPUs that can be used simultaneously. The second approach is to run the program by using cloud computing[23]. Opal web service[24] provides AutoDock on a cloud platform. Recently, Schrödinger tested virtual screening of 21 million compounds using their docking program on the Amazon cloud platform (Schrödinger website news). The docking calculation scaled from 0 up to about 50,000 steadily over three hours. The shortcoming of cloud computing is that it takes time to reach optimal scalability, which can result in a waste of CPU recourse at the beginning.

The third approach is to utilize GPU processors. Recently, large-scale GPU clusters and GPU/CPU hybrid clusters are gaining popularity in the scientific computing community[25]. However, their deployment and production use are associated with several challenges in terms of the application development process, job scheduling, and resource management. Yang and co-workers[26] have leveraged the computational power of GPUs to accelerate DOCK6's[27] Amber[28] scoring with the NVIDIA CUDA platform[29]. Their results show that a single GeForce 9800 GT GPU-based docking program achieves a 6.5x speedup with respect to the original version running on an AMD dual-core CPU. Korb and co-workers[25] developed a GPU-based protein-ligand docking program PLANTS, which has speedup factors up to 10x on a NVIDIA GeForce 8800 GTX GPU. Currently, most GPU-based docking programs use a single GPU processor so that their speedup factors are usually within 10, as compared to a single CPU. The fourth approach is to use a message passing interface (MPI) to parallelize the application for standard CPU clusters, which is still the mainstream of the high performance computing. An MPI-based implementation docking program based on Autodock 4 (Autodock4.lga.MPI) has been developed by Collignon et. al.[30]. It scales up to 8192 CPUs with a maximal overhead of 16.3%, two thirds of which are due to input/output operations, and one third originates from MPI operations. The relative high overhead is due to the usage of intermediate files. In 24 hours, on 8192 high-performance computing CPUs, 300K small flexible compounds or 11 million rigid compounds can be docked to a single rigid protein.

AutoDock Vina is a molecular docking program recently developed by Trott and Olson[31] that achieves an approximately two orders of magnitude speed-up compared with AutoDock 4,[21] developed at the same lab. Since its creation, AutoDock Vina has been widely deployed in molecular docking studies[32-35]. It also significantly improves the accuracy of the binding mode

predictions compared to AutoDock 4[36]. Further speed-up is achieved by using multithreading of docking pose generation on multicore CPU processor. AutoDock Vina calculates the grid maps on-the-fly, which makes it more user-friendly as compared to AutoDock 4. As a new generation of docking software from Olson's Lab, Vina uses a totally different sampling algorithm and scoring function from that of AutoDock 4. Vina employs a Monte Carlo based sampling algorithm named "iterated local search global optimizer" similar to the method developed by Abagyan et. al.[37], which is a succession of steps consisting of a mutation and then a local optimization step, with each step being accepted according to the Metropolis criterion. The Vina scoring function uses empirical terms that account for steric forces, attraction, repulsion, hydrophobicity, hydrogen bonding, and molecular flexibility, which allows rapid assessment of docking poses. Due to its high accuracy with fast computing speed, Vina is an ideal candidate code to be parallelized and run on HPC.

Here, we implemented an MPI and multithreading hybrid parallel scheme in the Vina molecular docking program. The resulting program is named VinaLC, where LC stands for Livermore Computing. The parallel performance of the VinaLC program was investigated on several different types of supercomputers, including petascale HPC machines. Furthermore, we evaluate VinaLC's accuracy by examining enrichment and pose quality. The DUD dataset was used to compare VinaLC with benchmark studies performed using other docking programs[38].

Computational Details

Mixed MPI and multithreading programming

In order to exploit the computing resource of HPC machines, the original code of Vina (version 1.1.2[31]) was modified using MPI and multithreading hybrid parallel programming. The original Vina is a multithreading C++ program using the Boost thread library[39], which runs on shared memory and is hard to scale up to larger numbers of CPUs. Nowadays, a typical supercomputer consists of a large number of nodes (1000s to 100,000s). Within a node several CPU cores share memory, and between the nodes the interconnection allows programs on each node to interact with each other in a distributed memory system, which is usually slower than within the node. Thus, a multithreading parallel program that uses distributed memory lacks efficiency when running on HPC. It will be more efficient to deploy a message-passing multiprocess. Therefore, we use a mixed MPI and multithreading scheme in the parallel program (Figure **1**), where within the node multithreading is used, and among different nodes an MPI parallel scheme is applied.

MPI Scheme

A careful design MPI scheme has been implemented in the VinaLC program so that it is able to achieve high scalability with low overhead. The calculating time for each docking calculation could be significantly different due to the differences among ligands, receptors, and the grid box sizes in the active site of receptor. Distributing docking calculations evenly on each MPI process will affect the MPI load balance because some processes will finish docking calculations early and then idle, waiting for other processes to complete their calculations. To keep every process busy and reduce the idling time, the program employs a master-slave MPI scheme as illustrated in the Figure 2. The master process is shown on the left side, and the slave processes are on the right. The MPI calls are designated by the solid-color boxes, paired in the same color, and

connected by the arrows. The directions of the arrows show the direction of the data flow for each pair of MPI calls. The light-grey boxes represent loops in the program. The master process has three ‘for loops’ and each slave process has one infinite ‘while loop’. For the master process, the first ‘for loop’ is a job loop, which goes through every combination of the receptor and ligand. Two pairs of MPI send/recv calls, colored in blue and green respectively, control the job flow. The free slave process sends its MPI rank to the master process. The master process tries to receive the rank of any free slave process by using MPI_Any_Source tag. If there are still docking calculations in the queue, the master process sends an unfinished job flag to the free slave process so that the infinite ‘while loop’ keeps running. All the input and output data are handled by the master process. The input data are packed into one data package so that only one pair of MPI send/recv calls is required to reduce the MPI overhead, so does output data. The master process sends the input data required for the docking calculation to the slave process. After receiving the input data, the slave process performs the docking calculation. The slave process sends the output data back to the master process when it finishes the assigned calculation from the master process. Only after the master process has assigned each slave process with a docking calculation will the master process start collecting the output data. The second ‘for loop’ is used to collect the data. If the size of the jobs is larger than the number of slave processes, the size of the loop equals the number of slave processes. If not, the size of the loop equals to the size of the jobs. The master process will garner the output data from any slave process by using MPI_Any_Source tag. Once the output data from the slave process is collected, then the master process will give that slave process another job. The third ‘for loop’ in the master process sends a finished job flag to free the slave processes. The slave process breaks the infinite ‘while loop’ after receive the job flag. The MPI run finishes after the third ‘for loop’. By implementing such a

master-slave MPI scheme, the master is in charge of job dispatching, input, and output while the slave processes are kept busy by running individual docking calculation until all the calculations are finished.

Code Implementation

The code implementation includes MPI parallel, input/output, data handling, and file formats. The original Vina code was converted to a single complex docking function, and an MPI program scaffold was built on top of this docking function. The MPI part of the code is in charge of the docking task dispatching, and the docking function performs individual docking calculations. The code was re-designed so that the master process will perform the input and output operations. The code for parsing input files was converted to a function so that it can be called by master process. The raw input and output data are stored in two C++ structs, respectively, so that only two pairs of MPI send/recv calls are required to communicate the data between the master and slave processes. The input data struct contains the information about the ligand and receptor, number of threads, docking grid, exhaustiveness of the conformer searching, and other options required for the docking calculation. The original Vina code parses the ligand and receptor PDBQT files and calculates the grids on the fly. Our parallel program has to handle thousands to millions ligand/receptor structures. To make it simple, the ligand and receptor structures are saved in several large multi-structure PDBQT files. For example, more than one million ZINC ligand structures can be saved in 40 multi-structure PDBQT files. Each PDBQT files contains about 25K ligands. Two ‘for loops’ are involved when the master process reads in the input structures. The outer loop loops through the receptor PDBQT file list, while the nested inner loop loops through the ligand PDBQT file list. After reading in one ligand structure, one docking calculation is launched on slave process by master. The master process reads in the

structure files of the ligand and receptor structures as a string stream and saves them in large character buffers/arrays as raw data in the input data struct. It also calculates the docking girds and stores them as arrays in the input data struct. The input data is passed from the master process to the designated slave process. The slave process will populate the grids from the input data and parses the raw ligand and receptor data into objects defined in VinaLC. The output data struct contains two character buffers to save the log and docking poses respectively. The original Vina logs are re-directed to a string stream by the slave process. The string stream is saved in the character buffer in the output data struct. Also, instead of writing an output file, the docking poses are piped into a string stream and then store in the character buffer in the output data struct. The output data struct is passed to the master process by the slave process. The master process saves the entire log and all docking poses into two GZip files to save the disk space. The GZip file for nine million docking poses is about 3 Gig bytes, which is easier to maintain than saving docking poses in millions of individual files. Many screen outputs that were in the original Vina program, such as progress bars, thread messages, job messages, etc, have been disabled to reduce the I/O activity.

VinaLC has also been further optimized from the original version in many other aspects. Computational expensive calculations have been replaced with cheaper and simpler algorithms without sacrificing accuracy or precision. For example, many "sqrt" functions, which calculate the square root of a number, have been removed from the code. The algorithms that involve these functions have been changed to alternate methods.

Benchmarking Data Sets

Benchmarking data sets have been carefully selected to carry out parallel performance analysis and docking accuracy validation. Two benchmarking data sets, ZINC[7] and DUD (a Directory of

Useful Decoys)[38] databases, were chosen in this study. ZINC is a free database of commercially available compounds for virtual screening. ZINC contains over 30 million purchasable compounds. We randomly selected 1 million compounds from the ZINC database to test the efficiency and scalability of VinaLC. The receptor for the ligand docking is the *Thermus thermophilus* gyrase B ATP-binding domain (PDB ID: 1KIJ). The grid box was generated at the center of the receptor active site with a cubic size of 106×106×106 with grid spacing of 0.375 Å. To benchmark the VinaLC docking accuracy calculation, the DUD database was selected because DUD is both a diverse and difficult database for structure-based virtual screening. DUD contains 2950 annotated ligands for 40 diverse targets and about 36 decoy molecules for each annotated ligand. A complete list of 40 targets and their full names can be found in the supporting information or the original DUD paper[38]. To avoid bias in the dataset, decoy molecules were assembled to physically resemble ligands but are chemically distinct from other molecules[38]. Thus, ligand enrichment is not simply a separation of gross features but rather reflects the ability to separate chemical features such that decoy molecules are unlikely binders. Although there are known issues and limitations within the DUD[40] database, the diversity of the set with the decoys challenges any docking code. DUD dataset, acting as a standard docking dataset, has been widely applied to benchmark various molecular docking programs[40-42]. In this study, DUD dataset was used to benchmark our VinaLC so that it can be compared with other programs. The ligand, decoy and receptor structures were obtained from the DUD website[43]. The SDF and PDB formats of the structure files were converted to PDBQT format using MGLTOOLS[44]. The grid box dimensions were set the same as that of Huang, et. al.[38]. The grid spacing was set to be 0.333 Å.

There are many ways to gauge the enrichment performance of the program. Enrichment factor (EF)[14,41,45] is one of methods that was used to measure the virtual screening performance of the VinaLC docking program.

$$EF^{x\%} = \frac{Actives_{sampled}}{Actives_{total}} \frac{N_{total}}{N_{sampled}},$$

where $actives_{sampled}$ is the number of actives found at x% of the screened database, $actives_{total}$ is the number of total actives in the database, $N_{sampled}$ is the number of compounds at x% of database, and N_{total} is the number of total compounds in the database. The enrichment factor has several deficiencies because it largely depends on the composition of the data set and is not stable at low x%. Thus, in this study we used the average value of EF calculated from 40 targets in the DUD data set in order to eliminate the variability of data composition and reduce the uncertainty of the value at low x%.

Input for Parallel VinaLC

The input for VinaLC employs a command-line style. The program will take the input parameters from either an input file or command line options. A typical command line to run VinaLC using the SLURM[46] job scheduler follows:

```
srun -N1284 –n1284 –c12 ./vinalc --recList recList.txt --ligList ligList.txt --geoList geoList.txt --exhaustiveness 12
```

This command line tells the SLURM job scheduler to run the calculation on 1284 nodes, launch 1284 processes simultaneously, and use 12 CPUs for each process. VinaLC reserves the MPI rank of 0 for the master process and treats the rest as slave process. The process automatically

detects the number of the CPUs and then launches the 12 threads for each docking calculation. VinaLC reads in lists (recList.txt and ligList.txt) for the receptor and ligand PDBQT files together with a list (geoList.txt) for the geometry of the receptor grid boxes. The option of "exhaustiveness" specifies the number of the Monte Carlo simulations to be run for each docking calculation and its default value is 8. Other options and their default values can be printed out by VinaLC if the user runs the program with option flag of "--help".

The VinaLC docking program has demonstrated its portability by running on various types of supercomputers. It has been ported to several different kinds HPC machines at Lawrence Livermore National Laboratory, including Linux clusters of Intel Xeon processors, Linux clusters of AMD Opteron processors, and a new IBM 20-petaflop supercomputer with BG/Q architecture[47]. The code was extensively tested on a Linux cluster of Intel Xeon processors for parallel processing, composed of 1,944 nodes, in which there are 1865 total compute nodes. Nodes are Intel Xeon 5660 dual-socket 6-core nodes, each with a total of 12 cores (2.8 GHz) and 24 GB of memory. Between the nodes, the InfiniBand QDR (quad data rate)[48] is the high-speed interconnect, which is necessary for the code to link to the MVAPICH library[49], a MPI library particularly tuned for the InfiniBand interconnect. The maximum number of nodes allowed for a batch job is 1284 nodes (15,408 cores).

Results and Discussion

Parallel VinaLC scales to a large set of cpus with low overhead

Parallel Scalability. The scalability of MPI and multithreading mixed parallel VinaLC program has been studied on a Linux cluster of Intel Xeon processors. The ligands in the test case were selected from the ZINC database, in which the first 100K compounds were docked into the active site of *Thermus thermophilus* gyrase B ATP-binding domain. The test case was calculated on 600, 1200, 2400, 6000, 12000, and 15408 CPU cores, respectively. The MPI wall time for the docking calculations decreases drastically as the CPU cores increases. As shown in Figure **3**a, the MPI wall time cuts by almost half each time the number of the CPU cores doubles. The average CPU time of each docking calculation only changes from 82.98 to 89.69 seconds, when number of CPU cores increases from 600 to 15408 (Figure **3**b). The trend line of the average CPU time deviates only slightly from the ideal average CPU time, as shown by dashed line. As shown in the Figure **3**c, the factors of speed up are almost identical to the ideal values when the number of CPU cores is less than 6000 while The factors of speed up are only slightly less than those of ideal ones when the number of CPU cores is larger than 6000. The average CPU time per docking run and the speed up plots both suggest that VinaLC scales very well up to more than 15K CPU cores. A test running with 1 million ZINC compounds docking in the active site of *Thermus thermophilus* gyrase B ATP-binding domain was completed in 1.4 hours on 15,408 CPU cores. By extrapolation, VinaLC would allow docking of about 17 million flexible compounds to a single protein in 24 hrs on 15K CPU cores.

MPI Performance analysis with mpiP. In order to probe the MPI performance, the VinaLC program was profiled using the mpiP library[50], a lightweight, scalable MPI profiling tool. Two metrics were calculated by mpiP at various numbers of CPU cores: 1) the MPI time percentages

for the master and slave processes and 2) the aggregate MPI time of the master and slave processes. The MPI time percentage is plotted against the number of the CPU cores in Figure **3**d. For each calculation there is only one master process and the rest of the large number of processes are slaves. Thus, the average MPI time percentage of both master and slave processes are close to that of the slave processes, which results in the red line almost overlapping with green one in Figure **3**d. With the increase of CPU cores, the master process must deal with more slave processes and handle the data output and input more frequently. The time percentage of the master process spent on the MPI operations decreases steadily, which is due to the increased burden on the master process, such as, compressing the output results, saving the docking poses, and logs. When the number of slave processes is small, the master process is more readily responsive to the requests of the slave processes. However, when the number of slave processes increases, the slave processes spend more time waiting for the master to allocate docking calculations. Therefore, the MPI time percentage of slave processes increases. The average MPI time percentage is measured to be 3.94% for 15408 CPU cores.

Considering the computing time for MPI activity alone, the aggregate MPI time was calculated at various numbers of CPU cores. The two MPI_recv calls colored in blue and green (Figure **2**), master process checking the available free slave process and slave process receiving a job flag from the master process, have accounted for more than 99% of aggregate MPI time of all 11 MPI send/recv calls. When the test case runs on 600 CPU cores, 83.78% of MPI time has been spent on the master process checking the available free slave processes, and 15.97% has been spent on the slave process receiving a job flag from the master process. The numbers are 55.32% (master checking free slave processes) and 44.33% (slave process receiving job flag) when the test case runs on 1200 CPU cores. As the number of CPU cores increases, the percentage of MPI time

spent on the master process checking the available free slave process decreases while the that of slave process receiving a job flag from the master process increases. When the test case runs on 15408 CPU cores, only 0.53% of MPI time has been spent on the master process checking the available free slave process, and 99.07% has been spent on the slave process receiving a job flag from the master process. The MPI_send calls for docking data input and output are colored in purple and orange, respectively, (Figure **2**) have totally accounted for less than 0.1% of MPI time although they have delivered almost 100% of the aggregate sent message. The MPI_send call for docking data output accounted for 90.76% of the aggregate sent message and that for docking data input is 9.24%. The use of the GZip format for output files has achieved a 7:1 compression ratio, which reduces the storage of results significantly.

Program profiling with Open|Speedshop. To investigate time consumption on each functionality in the code, VinaLC was run with Open|Speedshop (OpenSS) profiling tool[51]. Due to the overhead of OpenSS, VinaLC ran on only 60 CPU cores with a smaller test case. The program was first profiled with the "PC sampling experiment" option in order to determine CPU time spent on each function. VinaLC spends 94.11% on the actually docking calculations and about 5.52% on MPI operations. The function for pairwise energies and forces calculation in the Monte Carlo simulation is the most time-consuming step, which accounts for 31.89% of the calculation time. The second most time-consuming function is the 'docking grids evaluation' module that takes 14.40%. The function to carry out Hessian matrix calculation accounts for 4.34%. These results show that most of the computing time is spent on docking calculations. For rest of the functions, each takes less than 5% of computing time. The program was then run with the "IO" option to detect amount of time on I/O system calls. The program spends 96.47% time to read,

3.07% to write, and 0.46% to close files. The total time of I/O system calls can be neglected as compared to the total compute time.

Comparison to other parallel docking programs. Several other parallel docking programs have already been reported to run on high performance computing so that we can compare parallel performance of VinaLC to that of others. Compared to DOVIS, VinaLC can scale up to more than 15K CPU cores while DOVIS is limited to 256 CPU cores[22]. Our MPI parallel scheme is more efficient in communicating data and reduces the I/O activities so more CPU cores are exploited. Compared to the cloud computing, the MPI implementation reaches high scalability at the beginning of the calculations, utilizing the CPUs more efficiently, while it takes time for cloud computing to utilize all CPUs. Autodock4.lga.MPI[30] by Collignon et. al. uses a pure MPI parallel scheme. Although Autodock4.lga.MPI scales up to 8192 CPUs, it has an overhead of 16.3%[30]. VinaLC can scales very well to 15408 CPUs with an overhead of only 3.94%. This scale up translates into Autodock4.lga.MPI completing 300K small flexible compounds docking calculations on 8192 CPUs in 24 hours. In contrast, VinaLC can finish 17 million flexible compound docking calculations on 15408 CPUs in the same 24 hours. These advantages are achieved by using a mixed MPI and multithreading parallel scheme.

VinaLC accuracy is comparable with other docking programs.

X-ray ligand re-docking accuracy study. As mentioned in the method section, the DUD data set has been employed to validate the accuracy of VinaLC. One way to test a docking program is to perform the re-docking of X-ray structures. The X-ray ligands were taken out and then docked back to the active sites of the protein targets in the DUD data set. The RMSD values between the top scoring pose for a ligand and its X-ray conformation were calculated. The mean of RMSD

values of the top scoring poses is 2.76 Å with a standard deviation of 3.27 Å. The median value is 1.20 Å. A best pose with the smallest RMSD value in the top ten scoring poses was also determined. 10 out of 40 targets have the identical top scoring pose and best pose. The mean of RMSD values of the best poses is 2.03 Å with a standard deviation of 2.27 Å. The median value is 1.10 Å. The cutoff RMSD value for good and bad docking poses is 2.0 Å. Judging from the mean value and median would result in totally different conclusions. Evaluating the docking program by mean and median RMSD values apparently becomes an issue. For example, target ERantagonist (estrogen receptor antagonist) and COMT (catechol *O*-methyltransferase) have abnormally large RMSD values that exceed 10 Å. Such a few very poorly docked poses are the primary factor responsible for the large mean RMSD value.

Accuracy study with cumulative RMSD distribution plot. As shown in the previous section, a few abnormally large RMSD values will bring the bias to the mean and median values of RMSD. To avoid such bias, a cumulative RMSD distribution plot was used to demonstrate a complete picture of RMSD distribution (Figure **4**). Firstly, the RMSDs for all 40 targets were sorted in ascending order of RMSD values. The fractions of the complexes were calculated in a cumulative way with increasing of the RMSD values. The plot was generated with fraction of the complexes in the y-axis versus its RMSD value in the x-axis. A vertical dashed line was plotted at RMSD value of 2 Å to illustrate the cutoff value for the RMSD. As shown in the Figure **4**, 64.4% of the top scoring poses was identified with RMSD under the 2.0 Å cutoff while that for the best poses is 70.0%. For the best poses, all the targets have RMSD values within 10 Å and about half of the targets have RMSD values less than 1 Å. Overall, the VinaLC docking program performed well for re-docking of the X-ray ligand back into the active site of the X-ray structure with the default setting for the grid sizes and exhaustiveness (=8).

Comparison of cumulative distribution plots generated by different programs. The X-ray ligand re-docking accuracy of different docking programs can be compared using cumulative distribution plot. Cumulative distribution plots for a different data set with 68 protein-ligand complexes have been generated by Cross and co-workers[41]. Although they were using a completely different data set, the cumulative distribution plot still allows a relatively fair comparison of the VinaLC program from our study and the docking programs employed in their study because of the cumulative feature of the plots. We found that the performance of VinaLC in terms of cumulative distribution plot is better than the FlexX and PhDOCK, is comparable to the DOCK and Surflex, and is very close to the GLIDE.

The initial settings of the docking program are crucial to the docking accuracy. After visually inspecting the docking results for the target COMT, we found that the abnormally large RMSD value was due to the ligand being docked outside of the binding pocket. We tried to either reduce the grid sizes or enlarge the exhaustiveness value to allow more extensive conformational sampling. Both approaches reduced the RMSD values drastically. For example, using a smaller grid box with dimensions of 22×22×22 Å for target COMT reduces the RMSD value from 12.86 to 1.13 Å. Doubling the exhaustiveness value also reduces the RMSD value to 0.79 Å. In general, reducing the grid sizes will constrain the ligand to be docked within the active site and enlarging the exhaustiveness will increase the conformation searching time. Both result in obtaining significantly better poses. Thus, carefully defining the settings of the VinaLC docking program is crucial to obtaining successful results.

Docking enrichment performance. The Receiver Operating Characteristic (ROC) plot was employed in this study for measuring virtual screening performance[52-54]. The ROC method can effectively differentiate two populations so that it can be applied to differentiate the active

ligands against the non-active decoys. The plots of ROC curves for all 40 targets in the DUD data set are shown in Figure **5**. The ROC plots are arranged in sequence of 8 nuclear hormone receptors, 9 kinases, 3 serine proteases, 4 metalloenzymes, 2 folate enzymes, and 14 other targets[38], which are included in the DUD data set. The ROC plots are similar to the ligand enrichment plots[38]; however, the x-axis is the fraction of the decoy in the ROC curve plot rather than the percentage rank of the ligand in the whole data set combining ligand and decoy. For a target, the docking program can calculate the docking scores for both ligands and decoys. Both ligands and decoys were sorted according to the significance of docking scores. From high to low scores, one can calculate the fraction of a selected ligand by dividing its rank (within all ligands) by the total number of ligands ($I^{th}/N_{ligands}$). The fraction of a selected decoy was calculated in the same way as that of ligand. Then ligands and decoys were mixed and sorted again. If there are any ligands ranking ahead of a given decoy, the largest fraction number of these ligands is correspond to the fraction number of selected decoy. If not, zero corresponds to the fraction of the selected decoy. The ROC plot uses the fraction of each ligand and decoy as the y-axis and x-axis, respectively. The curve of ROC plot is always a monotonically increasing line.

The ROC plots for all 40 targets have revealed their docking enrichment performance. The AR (androgen receptor) target is the first ROC plot in Figure **5**. The red line is the staggered plot of the fraction of the selected active ligand versus that of selected decoy. If the docking program works well for a target, the docking scores for active ligands will be more significant than those of non-active decoys. In the plot, the fraction of the selected ligand will always be larger than its corresponding fraction of select decoy, and the red line in the plot will always be above and to the left of the black diagonal line. The plot demonstrates that VinaLC has good enrichment

performance for the AR target. The ROC plots show that VinaLC enriches the data of 28 out of 40 targets where most of the red line is above the diagonal line. Thus, VinaLC has good enrichment performance on 70% targets in the DUD data set. However, if the docking program prefers the decoys rather than the active ligands, the red line will be under the diagonal line. The ROC plots have shown that three targets, ACE (angiotensin-converting enzyme), AmpC (AmpC β-lactamase), and NA (neuraminidase), with the ROC curve under the diagonal lines. It indicates that VinaLC has bad enrichment performance in these targets. There are six targets, FGFr1 (fibroblast growth factor receptor kinase), PDGFrb (platelet derived growth factor receptor kinase), ADA (adenosine deaminase), GPB (glycogen phosphorylase β), HIVRt (HIV reverse transcriptase), and HMGA (hydroxymethylglutaryl-CoA reductase), have red lines almost overlapping with the diagonal lines, which means VinaLC has no preference in selecting ligands over decoys and thus the ranking of ligands is near a random selection. The rest of the plots are more complicate to interpret. For example, the COMT target has a high fraction of active ligands selected when there is a low fraction of decoys; however, a low fraction of active ligands are selected when there is a high fraction of decoys. This indicates that VinaLC has better enrichment performance when a low fraction of decoys exist. This is desirable outcome for early recovery of active ligands because people only care about a few top score docking poses (usually 0.5-5% of the whole database) during the virtual screening.

Virtual screening performance analysis. The Area Under the Curve (AUC) for the ROC is a crucial factor to evaluating the virtual screening performance. The value of AUC ranges from 0.0 to 1.0, in which 0.5 means random selection. In this study, the AUC values were categorized into four ranges. Out of a total of 40 targets, 8 targets have AUC values ranging from 0.75 to 1.0, which is considered significant enrichment. Twenty-two targets are in the range of 0.55 ~ 0.75,

which is considered good enrichment. Thus, about 75% targets (30 out of 40) in the DUD data set have enrichment. 8 targets are in the range of 0.45 ~ 0.55 which indicates random selection. Two targets (AmpC and NA) have enrichment worse than random selection with AUC values below 0.45. The mean ROC AUC of 40 DUD targets is 0.64 for the VinaLC program, which is better than DOCK, FlexX, ICM[37], and PhDOCK[55], but worse than GLIDE and Surflex[56,57] (Table **1**). The range of ROC AUC values is 0.60 to 068 at 95% confidence interval for VinaLC docking program.

Excellent early recovery of actives. Early recovery of actives is desirable for a docking program. The large AUC values are often obtained for the cases where early recovery of actives is good. However, the small AUC values do not necessarily mean that the early recovery of actives is not good. As pointed out in the previous section, for the COMT target, VinaLC actually has very good early recovery of actives although its AUC value are only 0.48, even below the random selection AUC value of 0.5. Thus, the mean ROC enrichment factors have been calculated at 0.5%, 1.0%, 2.0% 5.0% and 10% of the screening database (Table **1**). As expected the mean ROC enrichment decreases as the percentage of the screening database increases due to the monotonically increasing curve of the ROC plot. Compared to various docking programs, VinaLC has a fairly high mean ROC enrichments at low percentages. At 0.5 %, VinaLC has a mean ROC enrichment of 19.9 which is better than most the docking programs shown in the table except GLIDE SP and Surflex Ringflex. This indicates that VinaLC has very good early recovery of actives.

The DUD targets can be categorized into several enzyme families, including nuclear hormone receptors, kinases, serine proteases, folate enzymes, metalloenzymes, and other enzymes[38]. From the standpoint of enzyme families, VinaLC program has very good enrichment performance for

nuclear hormone receptors except targets PRs, serine proteases, and folate enzymes. For the kinases, three out of nine targets have near random selections. For metalloenzymes, the ADA target has near random selection. The remaining three metalloenzymes have good early recovery of actives.

Through the X-ray ligand re-docking and enrichment studies, the performance of VinaLC is demonstrated to be better than DOCK, FlexX, ICM, and PhDOCK, and is not as good as GLIDE and Surflex. Studies[31,36] have also shown that VinaLC has many advantages over AutoDock 4. Thus, in terms of accuracy the VinaLC docking program outperforms most of mainstream docking programs. Considering accuracy and speed, the parallel version of VinaLC program has many advantages over the other docking programs.

Conclusion

In this study, the original AutoDock Vina molecular docking program has been converted to a parallel VinaLC program using a hybrid scheme of MPI and multithreading. The MPI parallel implementation employed a master-slave approach to expedite docking calculation assignment from master to slave. The docking input/output data were packed to reduce the overhead of MPI activity. The output data was saved in GZip format, which has achieved high compression ratio to save the disk space. The program is portable to various types of HPC machines including petascale platforms. A parallel performance analysis of VinaLC program has shown that it can scale up to more than 15K CPU cores at very low overhead. The profiling of program has demonstrated that most computing time has been spent on the actual docking calculations. The percentage of I/O activity in the total computing time is negligible. The overall parallel

performance of VinaLC is much better than other parallel docking programs, such as, DOVIS, Autodock4.lga.MPI.

The accuracy of VinaLC docking program has been validated against DUD data set. The RMSD values between X-ray ligands and re-docking top poses for most targets in DUD data set are small. The ROC plots have demonstrated that VinaLC has good enrichment performance on most targets. An analysis on the mean ROC enrichment factors calculated at various percentage of the screening database indicates VinaLC has very good early recovery of actives. The mean ROC AUC has shown that VinaLC program is better than DOCK, FlexX, ICM, and PhDOCK, but not as good as GLIDE and Surflex. In summary, VinaLC docking program has outperformed most of mainstream docking programs in terms of docking accuracy.

The current MPI and multithreading hybrid parallel scheme has been successfully deployed on petascale supercomputers by using only one master node. The master node can handle all the input and output data on the petascale HPC machines. With explosive growth in biological data and computer power, we have expected such parallel scheme can be stretched to its limit in the future. If so, the single master parallel scheme can be easily converted to multiple masters, together with usage of the parallel I/O, which should be able to overcome such limitation when the exascale high performance computing is available.

Acknowledgement

The authors thank Scott Futral, John Gyllenhaal, and Ryan Day from Lawrence Livermore National Laboratory Computation Directorate for helpful discussion of the parallel scheme. We thank Livermore Computing for the computer time and Laboratory Directed Research and

Development for the funding (12-SI-004). This work was performed under the auspices of the United States Department of Energy by the Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Figures

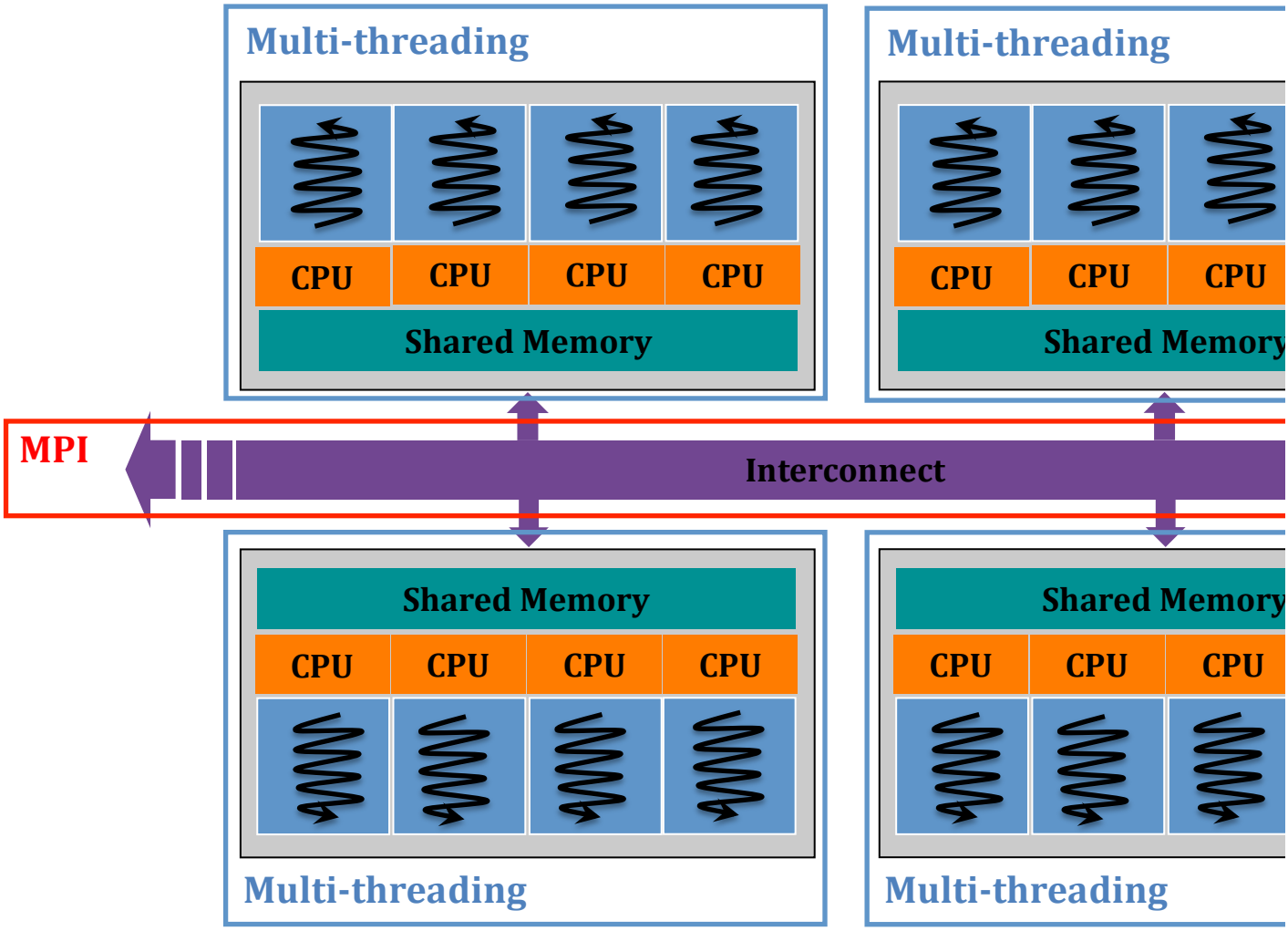


Figure 1 Mixed Parallel Scheme

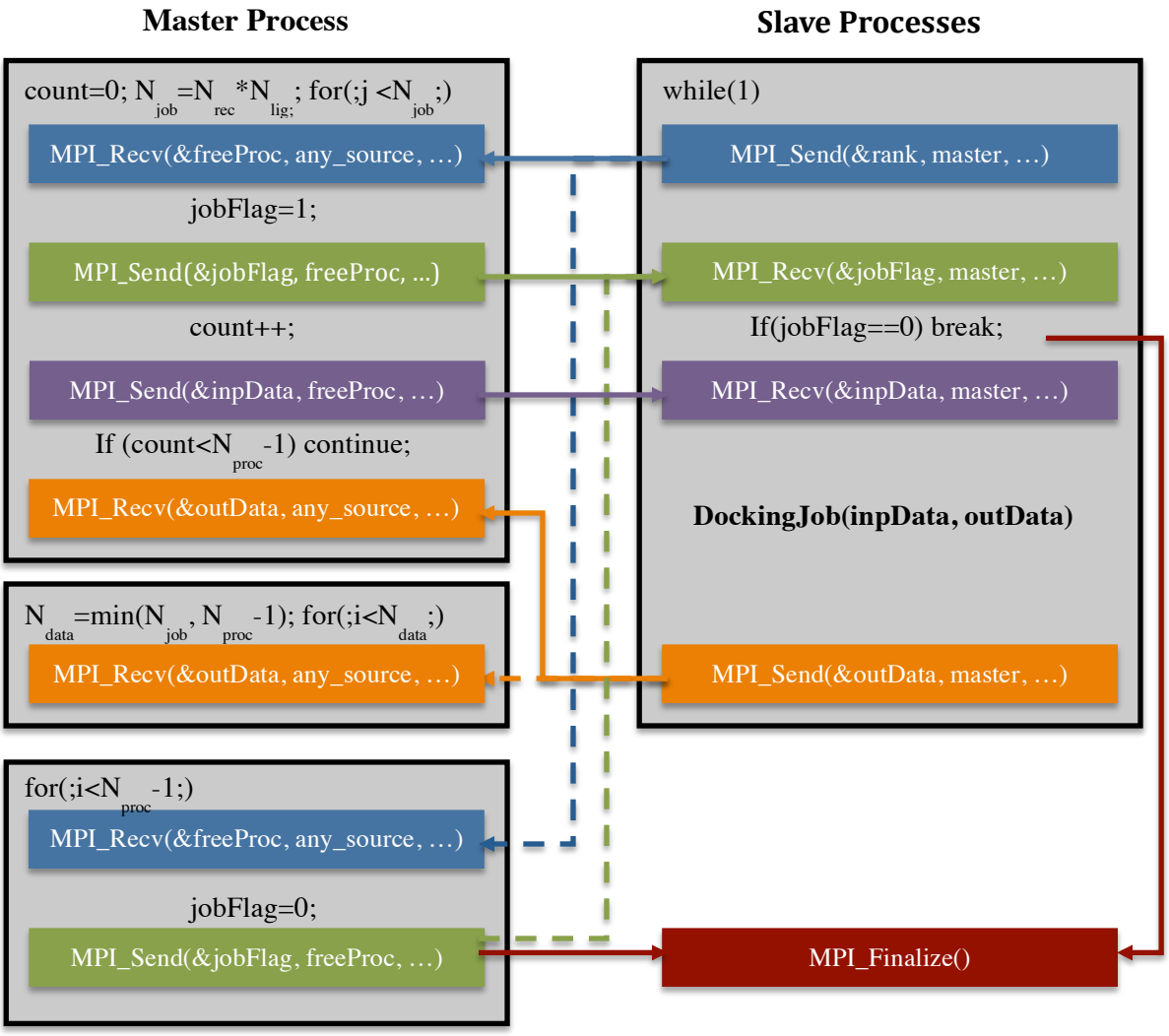


Figure 2 MPI Scheme. Pairs of MPI calls were labeled by different colors. Program loops were designated as black boxes with grey background. N_{rec}, N_{lig}, and N_{proc} are the number of the receptors, ligands, MPI processes, respectively.

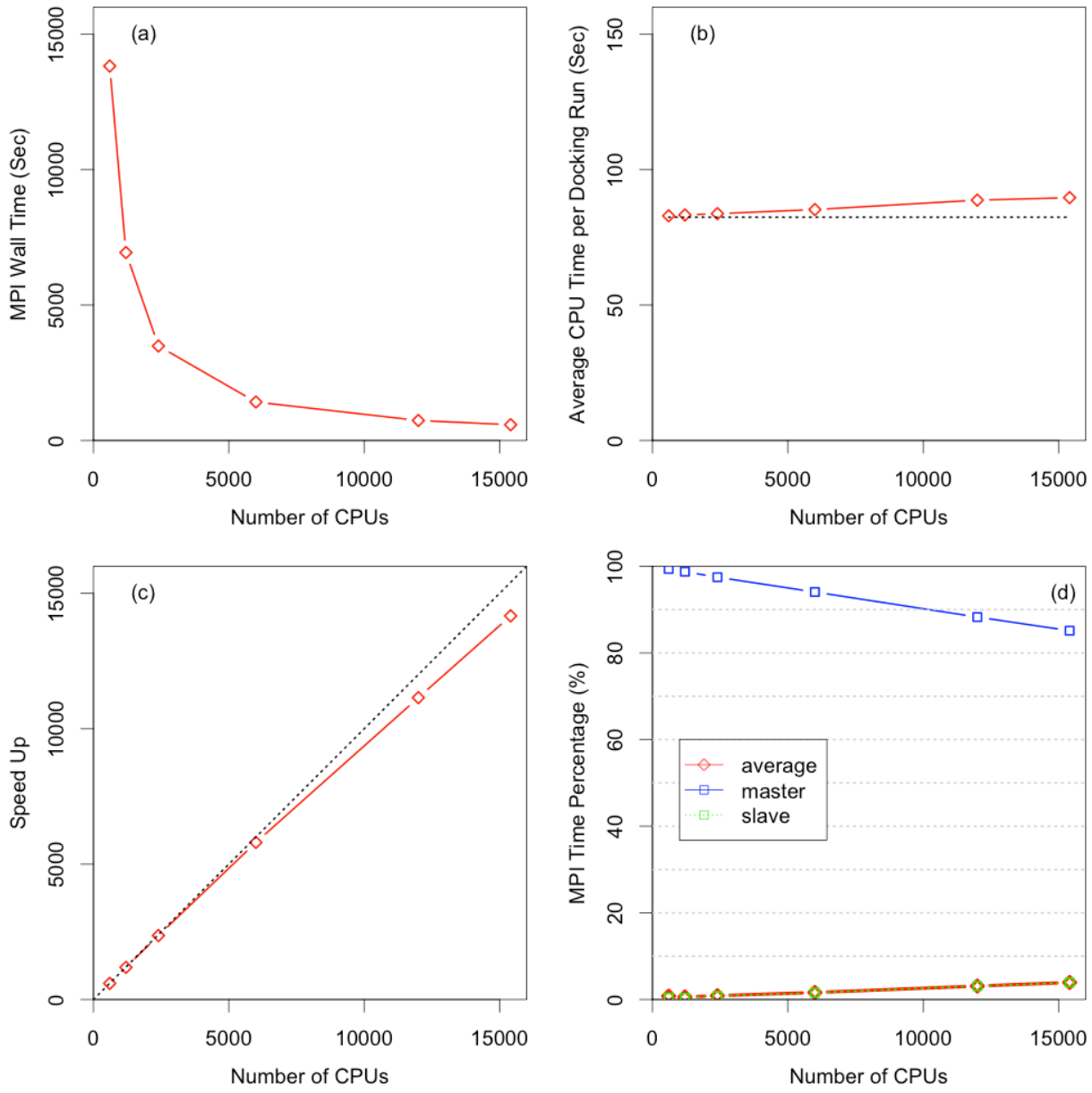


Figure 3 Parallel performance analysis of the VinaLC program on 600, 1200, 2400, 6000, 12000, and 15408 CPU cores. (a) MPI wall time versus number of the CPU cores. (b) Average CPU time per docking run versus number of the CPU cores. The dashed line is the ideal average CPU time per docking run. (c) The speed up of VinaLC docking calculation at different number of the CPU cores. The diagonal dashed line is the ideal scale-up. (d) The MPI time percentages for average, master, and slave processes at different number of CUP cores calculated by linking program to mpiP library. The test case for each simulation is a collection of 100K ligands selected from ZINC database, which were docked into active site of Thermus thermophilus gyrase B ATP-binding domain.

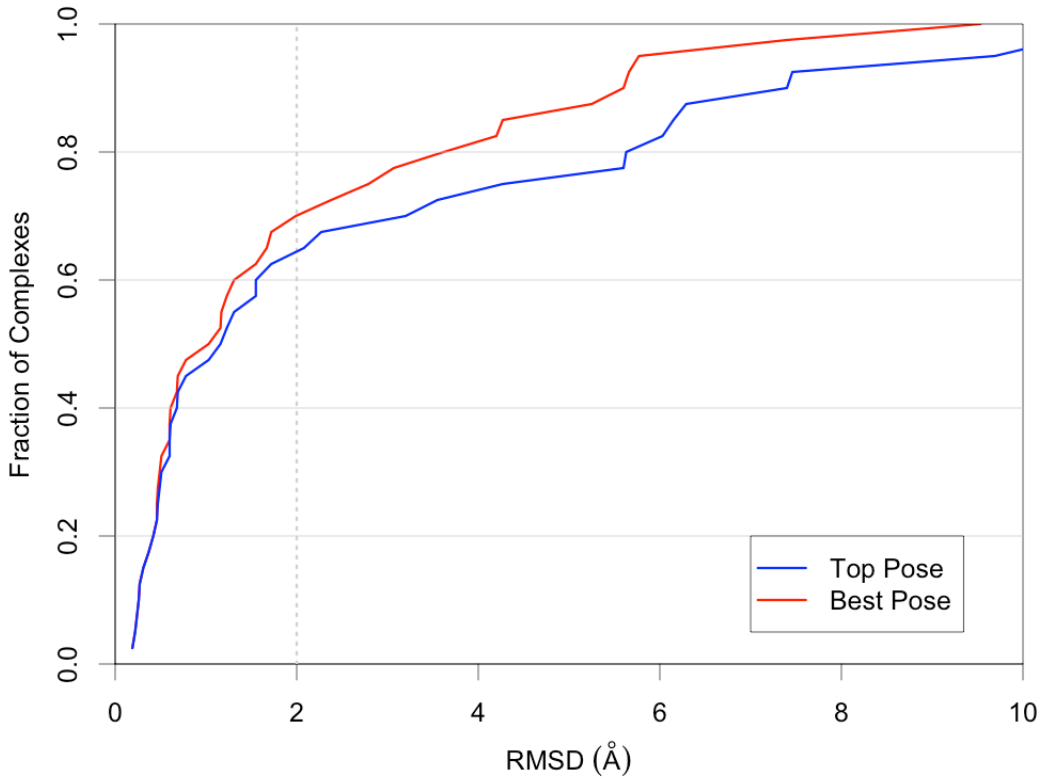
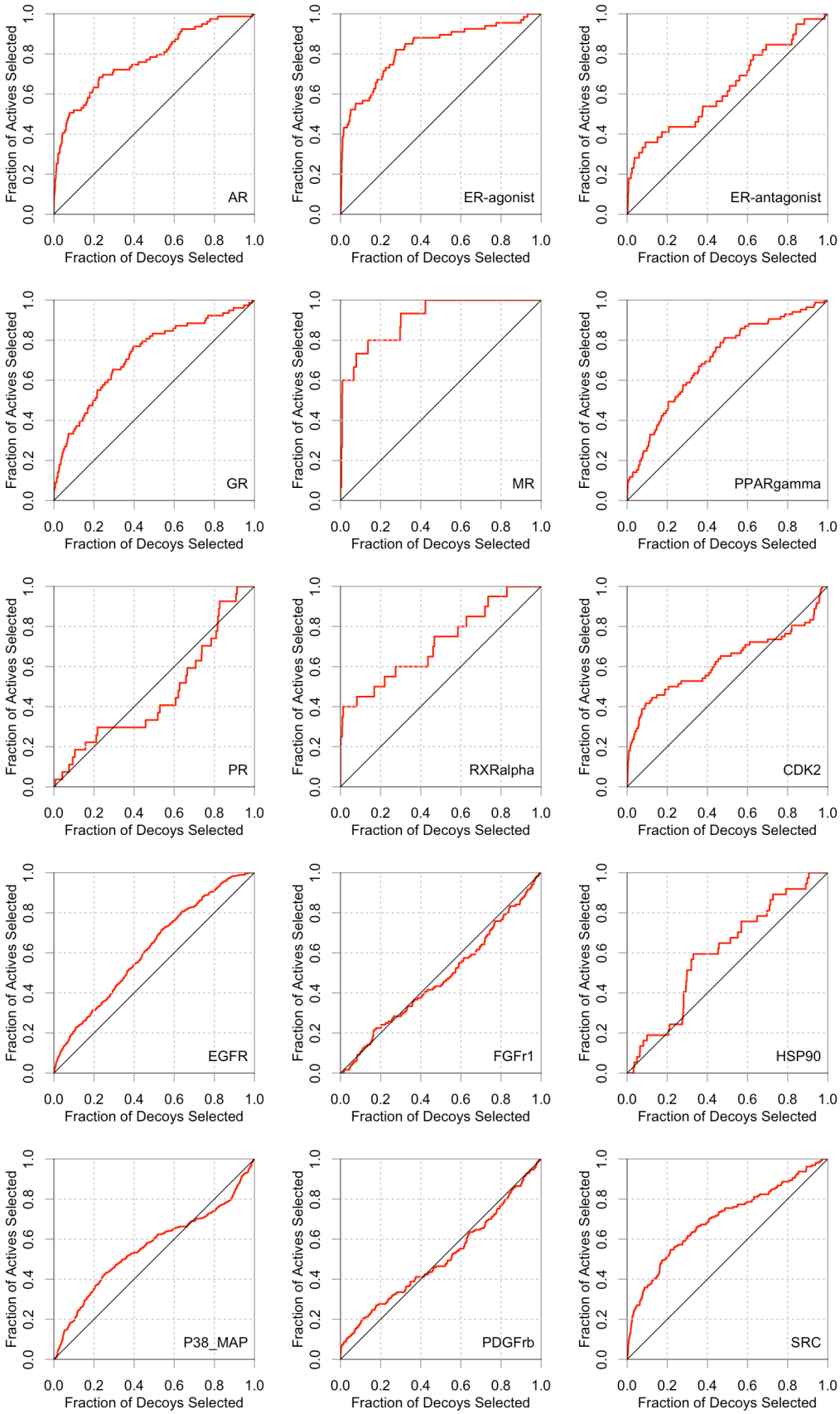
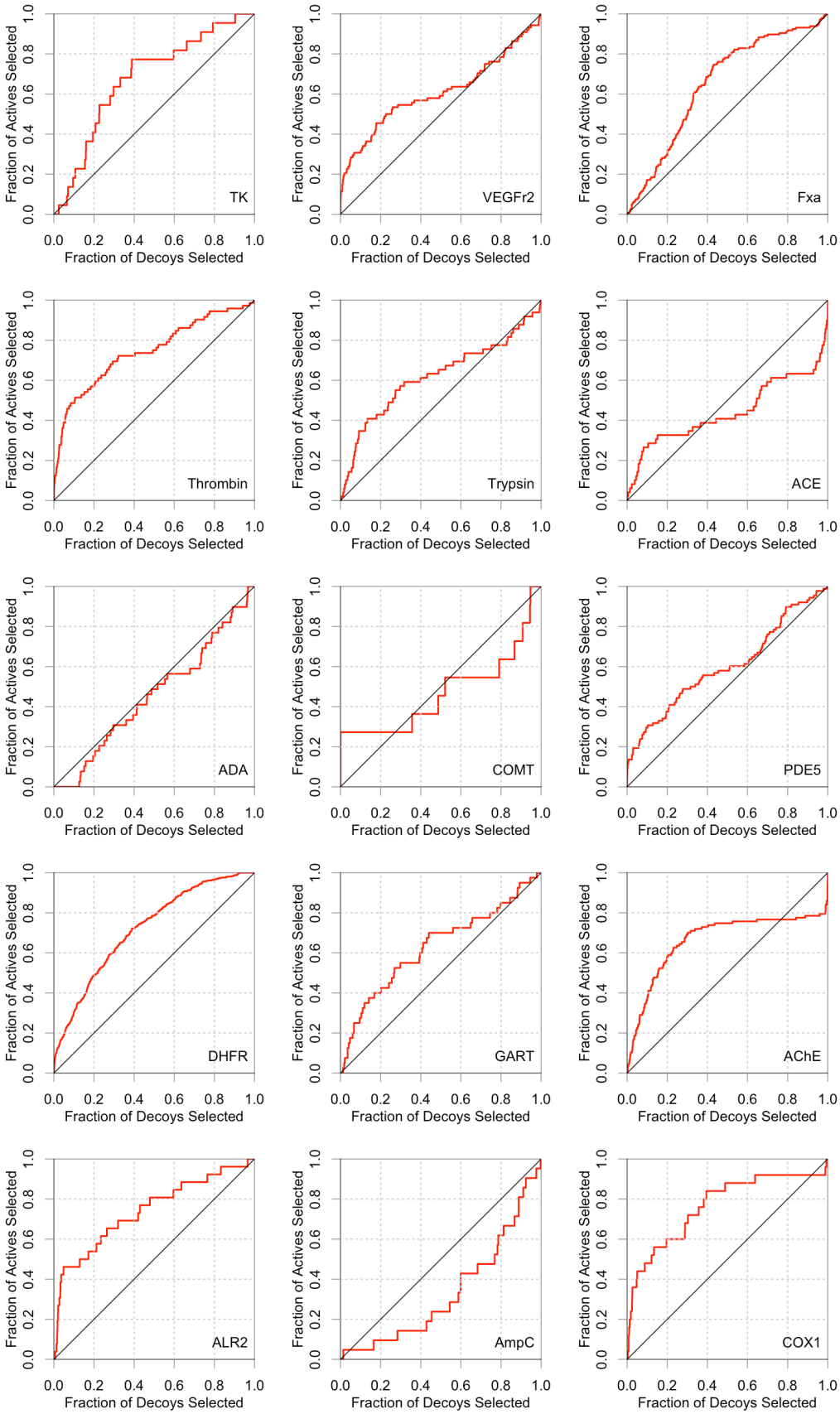


Figure 4 Cumulative RMSD distribution plot for the DUD data set. X-ray ligand conformation was used as input. The blue line is the RMSD between the top scoring pose and the X-ray structure while the red line is that of the best pose.


Fraction of Actives Selected
Fraction of Decoys Selected
0.0 0.2 0.4 0.6 0.8 1.0
AR
ER-agonist
ER-antagonist
GR
MR
PPARgamma
PR
RXRalpha
CDK2
EGFR
FGFr1
HSP90
P38_MAP
PDGFrb
SRC


TK
VEGFr2
Fxa
Thrombin
Trypsin
ACE
ADA
COMT
PDE5
DHFR
GART
AChE
ALR2
AmpC
COX1
Fraction of Actives Selected
Fraction of Decoys Selected
0.0
0.2
0.4
0.6
0.8
1.0

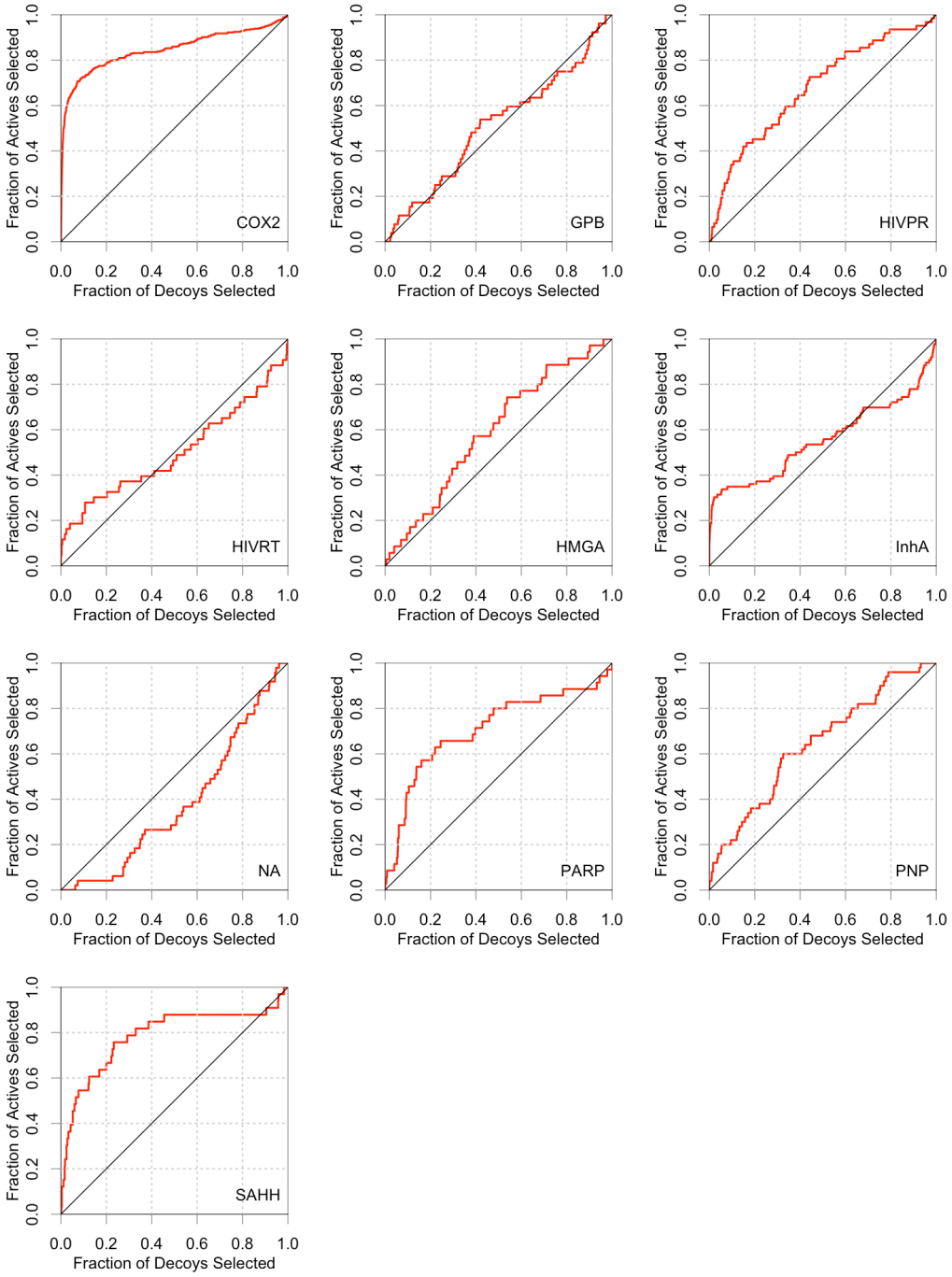


Figure 5 ROC plots for the 40 targets in the DUD data set. Diagonal lines indicate random performance.

Table 1 Statistical Results for Virtual Screening Using DUD data set.

Program	Source	Mean ROC AUC	95% CI	Mean ROC Enrichments				
				0.5%	1.0%	2.0%	5.0%	10.0%
DOCK	Ref[41]	0.55	0.50-0.59	18.8	12.3	8.2	4.7	3.0
FlexX	Ref[41]	0.61	0.54-0.67	13.7	9.8	7.2	4.4	3.1
GLIDE HTVS	Ref[41]	0.72	0.67-0.77	18.9	14.8	10.7	6.5	4.3
ICM	Ref[41]	0.63	0.58-0.68	16.9	12.7	8.0	4.6	3.1
PhDOCK	Ref[41]	0.59	0.54-0.64	16.9	11.3	7.7	4.1	2.8
Surflex	Ref[41]	0.66	0.61-0.70	14.3	11.1	7.9	4.9	3.4
GLIDE SP	Ref[41]	0.77	0.71-0.82	21.8	16.7	12.2	7.9	5.1
Surflex Ringflex	Ref[41]	0.72	0.67-0.77	20.0	16.2	12.0	6.8	4.3
VinaLC		0.64	0.60-0.68	19.9	11.4	9.0	6.1	3.2

Reference

1. Jorgensen, W. L. The many roles of computation in drug discovery. *Science* 2004, 303(5665), 1813.
2. Kitchen, D. B.; Decornez, H.; Furr, J. R.; Bajorath, J. Docking and scoring in virtual screening for drug discovery: methods and applications. *Nat Rev Drug Discov* 2004, 3(11), 935.
3. Cheng, T. J.; Li, Q. L.; Zhou, Z. G.; Wang, Y. L.; Bryant, S. H. Structure-Based Virtual Screening for Drug Discovery: a Problem-Centric Review. *Aaps Journal* 2012, 14(1), 133.
4. Hopkins, A. L.; Groom, C. R. The druggable genome. *Nature Reviews Drug Discovery* 2002, 1(9), 727.
5. Berman, H. M., et al. The Protein Data Bank. *Nucleic Acids Res* 2000, 28(1), 235.
6. Kirkpatrick, P.; Ellis, C. Chemical space. *Nature* 2004, 432(7019), 823.
7. Irwin, J. J.; Shoichet, B. K. ZINC - A free database of commercially available compounds for virtual screening. *J Chem Inf Model* 2005, 45(1), 177.
8. Ewing, T. J. A.; Makino, S.; Skillman, A. G.; Kuntz, I. D. DOCK 4.0: Search strategies for automated molecular docking of flexible molecule databases. *J Comput Aided Mol Des* 2001, 15(5), 411.
9. Meng, E. C.; Shoichet, B. K.; Kuntz, I. D. Automated docking with grid‐based energy evaluation. *J Comput Chem* 1992, 13(4), 505.
10. Goodsell, D. S.; Morris, G. M.; Olson, A. J. Automated docking of flexible ligands: Applications of AutoDock. *J Mol Recognit* 1996, 9(1), 1.
11. Morris, G. M., et al. Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function. *J Comput Chem* 1998, 19(14), 1639.
12. Rarey, M.; Kramer, B.; Lengauer, T.; Klebe, G. A fast flexible docking method using an incremental construction algorithm. *J Mol Biol* 1996, 261(3), 470.
13. Friesner, R. A., et al. Glide: A new approach for rapid, accurate docking and scoring. 1. Method and assessment of docking accuracy. *J Med Chem* 2004, 47(7), 1739.
14. Halgren, T. A., et al. Glide: A new approach for rapid, accurate docking and scoring. 2. Enrichment factors in database screening. *J Med Chem* 2004, 47(7), 1750.
15. Jones, G.; Willett, P.; Glen, R. C.; Leach, A. R.; Taylor, R. Development and validation of a genetic algorithm for flexible docking. *J Mol Biol* 1997, 267(3), 727.
16. Halperin, I.; Ma, B. Y.; Wolfson, H.; Nussinov, R. Principles of docking: An overview of search algorithms and a guide to scoring functions. *Proteins-Structure Function and Genetics* 2002, 47(4), 409.
17. Warren, G. L., et al. A critical assessment of docking programs and scoring functions. *J Med Chem* 2006, 49(20), 5912.
18. Wang, R. X.; Lu, Y. P.; Wang, S. M. Comparative evaluation of 11 scoring functions for molecular docking. *J Med Chem* 2003, 46(12), 2287.
19. Raha, K., et al. The role of quantum mechanics in structure-based drug design. *Drug Discovery Today* 2007, 12(17-18), 725.
20. *PBS Professional home page*, http://www.pbsworks.com/.

21. Morris, G. M., et al. AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility. *J Comput Chem* 2009, 30(16), 2785.
22. Jiang, X. H.; Kumar, K.; Hu, X.; Wallqvist, A.; Reifman, J. DOVIS 2.0: an efficient and easy to use parallel virtual screening tool based on AutoDock 4.0. *Chemistry Central Journal* 2008, 2.
23. Garg, V.; Arora, S.; Gupta, C. Cloud Computing Approaches to Accelerate Drug Discovery Value Chain. *Combinatorial Chem High Throughput Screening* 2011, 14(10), 861.
24. Ren, J. Y.; Williams, N.; Clementi, L.; Krishnan, S.; Li, W. W. Opal web services for biomedical applications. *Nucleic Acids Res* 2010, 38, W724.
25. Korb, O.; Stutzle, T.; Exner, T. E. Accelerating Molecular Docking Calculations Using Graphics Processing Units. *J Chem Inf Model* 2011, 51(4), 865.
26. Yang, H. L., et al. In Advances in Computational Biology; Arabnia, H. R., Ed.; *Springer-Verlag Berlin*: Berlin, 2010, p 497.
27. Lang, P. T., et al. DOCK 6: Combining techniques to model RNA-small molecule complexes. *RNA-Publ RNA Soc* 2009, 15(6), 1219.
28. Wang, J. M.; Wolf, R. M.; Caldwell, J. W.; Kollman, P. A.; Case, D. A. Development and testing of a general amber force field. *J Comput Chem* 2004, 25(9), 1157.
29. *CUDA™ is a parallel computing platform and programming model invented by NVIDIA*, http://www.nvidia.com/object/cuda_home_new.html.
30. Collignon, B.; Schulz, R.; Smith, J. C.; Baudry, J. Task-parallel message passing interface implementation of Autodock4 for docking of very large databases of compounds using high-performance super-computers. *J Comput Chem* 2011, 32(6), 1202.
31. Trott, O.; Olson, A. J. AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *J Comput Chem* 2010, 31(2), 455.
32. Seeliger, D.; de Groot, B. L. Ligand docking and binding site analysis with PyMOL and Autodock/Vina. *J Comput Aided Mol Des* 2010, 24(5), 417.
33. Ranjan, N.; Andreasen, K. F.; Kumar, S.; Hyde-Volpe, D.; Arya, D. P. Aminoglycoside Binding to Oxytricha nova Telomeric DNA. *Biochemistry (Mosc)* 2010, 49(45), 9891.
34. Sepe, V., et al. Discovery of Sulfated Sterols from Marine Invertebrates as a New Class of Marine Natural Antagonists of Farnesoid-X-Receptor. *J Med Chem* 2011, 54(5), 1314.
35. Plaza, A.; Keffer, J. L.; Bifulco, G.; Lloyd, J. R.; Bewley, C. A. Chrysophaentins A-H, Antibacterial Bisdiarylbutene Macrocycles That Inhibit the Bacterial Cell Division Protein FtsZ. *J Am Chem Soc* 2010, 132(26), 9069.
36. Chang, M. W.; Ayeni, C.; Breuer, S.; Torbett, B. E. Virtual Screening for HIV Protease Inhibitors: A Comparison of AutoDock 4 and Vina. *PLoS One* 2010, 5(8).
37. Abagyan, R.; Totrov, M.; Kuznetsov, D. ICM - A NEW METHOD FOR PROTEIN MODELING AND DESIGN - APPLICATIONS TO DOCKING AND STRUCTURE PREDICTION FROM THE DISTORTED NATIVE CONFORMATION. *J Comput Chem* 1994, 15(5), 488.
38. Huang, N.; Shoichet, B. K.; Irwin, J. J. Benchmarking sets for molecular docking. *J Med Chem* 2006, 49(23), 6789.
39. *Boost Organization Boost C++ Libraries is available at:* , http://www.boost.org/.
40. Good, A. C.; Oprea, T. I. Optimization of CAMD techniques 3. Virtual screening enrichment studies: a help or hindrance in tool selection? *J Comput Aided Mol Des* 2008, 22(3-4), 169.

41. Cross, J. B., et al. Comparison of Several Molecular Docking Programs: Pose Prediction and Virtual Screening Accuracy. *J Chem Inf Model* 2009, 49(6), 1455.
42. von Korff, M.; Freyss, J.; Sander, T. Comparison of Ligand- and Structure-Based Virtual Screening on the DUD Data Set. *J Chem Inf Model* 2009, 49(2), 209.
43. *DUD database, A Directory of Useful Decoys, Ligand, decoy, and target structures available at:* , http://dud.docking.org/.
44. Sanner, M. F. Python: A programming language for software integration and development. *J Mol Graph Model* 1999, 17(1), 57.
45. Pearlman, D. A.; Charifson, P. S. Improved scoring of ligand-protein interactions using OWFEG free energy grids. *J Med Chem* 2001, 44(4), 502.
46. SLURM. *Simple Linux Utility for Resource Management*, https://computing.llnl.gov/linux/slurm/.
47. *Sequoia, the new IBM 20-petaflops supercomputing system at LLNL*, https://asc.llnl.gov/computing_resources/sequoia/.
48. *InfiniBand is a switched fabric communications link used in high-performance computing and enterprise data centers*, http://www.infinibandta.org/.
49. *MVAPICH: MPI over InfiniBand, 10GigE/iWARP and RoCE*, http://mvapich.cse.ohio-state.edu/.
50. *mpiP: Lightweight, Scalable MPI Profiling*, http://mpip.sourceforge.net/.
51. *Open|SpeedShop, an open source multi platform Linux performance tool* http://www.openspeedshop.org.
52. Swets, J. A.; Dawes, R. M.; Monahan, J. Better decisions through science. *Sci Am* 2000, 283(4), 82.
53. Kellenberger, E.; Foata, N.; Rognan, D. Ranking targets in structure-based virtual screening of three-dimensional protein libraries: Methods and problems. *J Chem Inf Model* 2008, 48(5), 1014.
54. Jain, A. N. Morphological similarity: A 3D molecular similarity method correlated with protein-ligand recognition. *J Comput Aided Mol Des* 2000, 14(2), 199.
55. Joseph-McCarthy, D.; Alvarez, J. C. Automated generation of MCSS-derived pharmacophoric DOCK site points for searching multiconformation databases. *Proteins-Structure Function and Genetics* 2003, 51(2), 189.
56. Jain, A. N. Surflex: Fully automatic flexible molecular docking using a molecular similarity-based search engine. *J Med Chem* 2003, 46(4), 499.
57. Jain, A. N. Surflex-Dock 2.1: Robust performance from ligand energetic modeling, ring flexibility, and knowledge-based search. *J Comput Aided Mol Des* 2007, 21(5), 281.

Supporting Information

Table S1: List of DUD targets and RMSD of top scoring poses and best poses.

				Vina Docking Pose RMSD (Å)	
Protein	PDB ID	Ligands #	Decoys #	Top scoring pose	Best pose
AR	1xq2	74	2630	0.47	0.47
ERagonist	1l2i	67	2361	0.60	0.60
ERantagonist	3ert	39	1399	10.43	9.53
GR	1m2z	78	2804	0.31	0.31
MR	2aa2	15	535	0.26	0.26
PPARg	1fm9	81	2910	3.2	2.38
PR	1sr7	27	967	0.49	0.49
RXRa	1mvc	20	708	0.22	0.22
CDK2	1ckp	50	1780	7.46	3.62
EGFr	1m17	416	14914	1.55	1.55
FGFr1	1agw	11	4216	1.03	1.03
HSP90	1uy6	24	861	0.68	0.68
P38 MAP	1kv2	234	8399	0.60	0.60
PDGFrb	model	157	5625	0.61	0.61
SRC	2src	162	5801	6.15	5.66
TK	1kim	22	785	5.63	5.25
VEGFr2	1vr2	74	2647	1.23	1.23
FXa	1f0r	142	5102	1.55	0.46
thrombin	1ba8	65	2294	2.27	1.99
trypsin	1bju	43	1545	0.69	0.69
ACE	1o86	49	1728	7.40	7.40
ADA	1stw	23	822	1.72	1.72
COMT	1h1d	12	430	12.81	3.07
PDE5	1xp0	51	1810	0.78	0.78
DHFR	3dfr	201	7150	3.55	2.79
GART	1c2t	21	753	2.08	1.17
AChE	1eve	105	3732	1.16	1.16
ALR2	1ah3	26	920	6.29	5.77
AmpC	1xgj	21	734	6.03	1.67
COX-1	1q4g	25	850	0.42	0.42
COX-2	1cx2	349	12491	5.60	5.60
GPB	1a8i	52	1851	0.24	0.24
HIVPR	1hpx	53	1888	9.69	4.20
HIVRT	1rt1	40	1439	4.27	4.27
HMGR	1hw8	35	1242	1.31	1.31
InhA	1p44	85	3043	0.27	0.27
NA	1a4g	49	1745	0.46	0.46
PARP	1efy	33	1178	0.51	0.51

PNP	1b8o	25	884	0.19	0.19
SAHH	1a7a	33	1159	0.37	0.37

* Abbreviations: ACE, angiotensin-converting enzyme; AChE, acetylcholinesterase; ADA, adenosine deaminase; ALR2, aldose reductase; AmpC, AmpC β-lactamase; AR, androgen receptor; CDK2, cyclindependent kinase 2; COMT, catechol *O*-methyltransferase; COX-1, cyclooxygenase-1; COX-2, cyclooxygenase-2; DHFR, dihydrofolate reductase; EGFr, epidermal growth factor receptor; ER, estrogen receptor; FGFr1, fibroblast growth factor receptor kinase; FXa, factor Xa; GART, glycinamide ribonucleotide transformylase; GPB, glycogen phosphorylase β; GR, glucocorticoid receptor; HIVPR, HIV protease; HIVRT, HIV reverse transcriptase; HMGR, hydroxymethylglutaryl-CoA reductase; HSP90, hu- man heat shock protein 90; InhA, enoyl ACP reductase; MR, mineralocorticoid receptor; NA, neuraminidase; P38 MAP, P38 mitogen activated protein; PARP, poly(ADP-ribose) polymerase; PDE5, phosphodiesterase 5; PDGFrb, platelet derived growth factor receptor kinase; PNP, purine nucleoside phosphorylase; PPARg, peroxisome proliferator activated receptor γ; PR, progesterone receptor; RXRa, retinoic X receptor α; SAHH, S-adenosyl-homocysteine hydrolase; SRC, tyrosine kinase SRC; TK, thymidine kinase; VEGFr2, vascular endothelial growth factor receptor; ATP, adenosine-5'-triphosphate; β-GAR, β-glycinamide ribonucleotide; NAD(P)-(H), nicotinamide adenine dinucleotide (phosphate)-(reduced); PLP, pyridoxal-5'-phosphate.